*Fundamenta Informaticae XX (2009) 1–19*

*IOS Press*

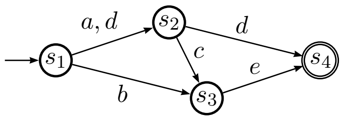
# Series-Parallel Automata and Short Regular Expressions

**Nelma Moreira**[*] [C] **and Rogério Reis**[*]
*DCC-FC & LIACC, Universidade do Porto*
*R. do Campo Alegre 1021/1055, 4169-007 Porto, Portugal*
{*nam,rvr*}*@ncc.up.pt*

**Abstract.** Computing short regular expressions equivalent to a given finite automaton is a hard task. In this work we present a class of acyclic automata for which it is possible to obtain in time $\mathcal{O}(n^2 \log n)$ an equivalent regular expression of size $\mathcal{O}(n)$. A characterisation of this class is made using properties of the underlying digraphs that correspond to the series-parallel digraphs class. Using this characterisation we present an algorithm for the generation of automata of this class and an enumerative formula for the underlying digraphs with a given number of vertices.

## 1. Introduction

Computing a regular expression from a given finite automaton can be achieved by well-known algorithms based on Kleene's theorem [20], establishing the equivalence between languages accepted by finite automata and languages represented by regular expressions. However the resulting regular expression depends on the order in which the automaton's states are considered in the conversion. In particular, this is the case if the algorithm is based on the *state elimination algorithm* [35]. Consider, for example, the following automaton:




If we remove the state $s_2$ and then the state $s_3$, the expression $(a + d)d + ((a + d)c + b)e$ is obtained. But if we remove first $s_3$ and then $s_2$, we obtain the regular expression $be + (a + d)(ce + d)$. In the first

[*]This work was partially funded by FCT and Program POSI.
[C]Corresponding author

case, the expression $a + d$ occurs two times, and in the second, the symbol $e$ occurs two times. The first case corresponds to the application of the distributivity rule on the right, and the second to the application of that rule on the left. In this last case the resulting expression has less symbols than the first one. If our goal is to obtain an equivalent regular expression with short size from a given automaton, the order in which we consider the automaton states is of great importance. Moreover, given an automaton with $n$ states and $k$ alphabetic symbols the upper bound for the size of the equivalent regular expression is $\mathcal{O}(nk4^n)$ (and no general smaller lower bounds are known) [18, 6]. The problem of obtaining a minimal regular expression equivalent to a given automaton is PSPACE-complete and NP-complete for acyclic automata [19]. If an unary regular language ($k = 1$) is accepted by a (non)deterministic automaton with $n$ states there exists an equivalent regular expression of size ($\mathcal{O}(n^2)$) $\mathcal{O}(n)$, respectively [6].

In this work we characterise a class of acyclic automata for which it is easy to find an order of state elimination such that the resulting regular expressions have size linear in the number of the automata transitions. The characterisation of this class is made using properties of the underlying digraphs that correspond to the series-parallel digraphs class.

The work reported in this paper was partially presented in Morais *et al.* [25, 26] and Reis [27], and is organised as follows. In the next section, we review some basic notions and introduce notation used in this paper. Section 3 defines the class of series-parallel (SP) automata in terms of the underlying digraphs and shows that it is possible to compute a linear size regular expression from an SP automaton. Section 4 presents an efficient algorithm for determining if an automaton is SP, and if it is the case shows how to obtain the correspondent short regular expression. Section 5 introduces another characterisation of SP digraphs that is used to give an enumerative formula. Some related work is discussed in Section 6 and Section 7 concludes.

## 2. Preliminaries

We recall the basics of digraphs, finite automata and regular expressions that can be found in standard books [18, 16, 1]. A digraph $D = (V, E)$ consists of a finite set $V$ of vertices and a set $E$ of ordered pairs of vertices, called $arcs$. If $(u, v)$ in $E$, $u$ is *adjacent to* (or incident to) $v$ and $v$ is *adjacent from* $u$. For each vertex $v$, the *indegree* of $v$ is the number $n_i$ of vertices adjacent to it and the *outdegree* of $v$ is the number $n_o$ of vertices adjacent from it, and we write $v(n_i; n_o)$. An arc $(u, v)$ can be denoted by $uv$. A *path* between $v_0$ and $v_n$ is a sequence $v_0v_1, v_1v_2, \ldots, v_{n-1}v_n$ of arcs, and is denoted by $\overline{v_0 \cdots v_n}$, or $\overline{v_0 \cdots v_k \cdots v_n}$, for $1 \leq k < n$. A path is *simple* if all the vertices in it are distinct. The length of a path is the number of arcs in the path. A path is a *cycle* if $v_0 = v_n$ and $n \geq 1$. A digraph that has no cycles is called *acyclic*. For an acyclic digraph $D = (V, E)$, there is a *topological* ordering $o$ of its vertices, *i.e.*, such that if $(u, v) \in E$ then $o(u) < o(v)$.

An alphabet $\Sigma$ is a nonempty set of symbols. A string over an alphabet $\Sigma$ is a finite sequence of symbols of $\Sigma$. The empty string is denoted by $\epsilon$. The set $\Sigma^\star$ is the set of all strings over $\Sigma$. A language $L$ is subset of $\Sigma^\star$. If $L_1$ and $L_2$ are two languages, then $L_1 \cdot L_2 = \{xy \mid x \in L_1 \text{ and } y \in L_2\}$. The operator $\cdot$ is often omitted. A regular expression (r.e.) $r$ over $\Sigma$ represents a (regular) language $L(r) \subseteq \Sigma^\star$ and is inductively defined by: $\emptyset$ is a r.e and $L(\emptyset) = \emptyset$; $\epsilon$ is a r.e and $L(\epsilon) = \{\epsilon\}$; $a \in \Sigma$ is a r.e and $L(a) = \{a\}$; if $r_1$ and $r_2$ are r.e., $(r_1 + r_2)$, $(r_1r_2)$ and $(r_1)^\star$ are r.e., respectively with $L((r_1 + r_2)) = L(r_1) \cup L(r_2)$, $L((r_1r_2)) = L(r_1)L(r_2)$ and $L((r_1)^\star) = L(r_1)^\star$. We adopt the usual convention that $\star$ has precedence over $\cdot$, and $\cdot$ has higher priority than $+$, and we omit outer parentheses. Let $R_\Sigma$ be the set of regular expressions over $\Sigma$. Two regular expressions $r_1$ and $r_2$ are equivalent if $L(r_1) = L(r_2)$, and we write $r_1 \equiv r_2$. In this work, we will take the *size* of a regular expression $r$ to be the number of symbols from $\Sigma$ contained in $r$, and we denote it by $|r|$.

A *nondeterministic finite automaton* (NFA) $A$ is a quintuple $(S, \Sigma, \delta, s_0, F)$ where $S$ is finite set of states, $\Sigma$ is the alphabet, $\delta \subseteq S \times \Sigma \cup \{\epsilon\} \times S$ the transition relation, $s_0$ the initial state and $F \subseteq S$ the set of final states. An NFA without $\epsilon$-transitions is *deterministic* (DFA) if for each pair $(s, a) \in S \times \Sigma$ there exists at most one $s'$ such that $(s, a, s') \in \delta$. For $s \in S$ and $a \in \Sigma$, we denote by $\delta(s, a) = \{p \mid (s, a, p) \in \delta\}$, and we can extend this notation to $x \in \Sigma^\star$, by $\delta(s, ax) = \delta(\delta(s, a), x)$. The *language* accepted by $A$ is $L(A) = \{x \in \Sigma^\star \mid \delta(s_0, x) \cap F \neq \emptyset\}$. Two NFA are *equivalent* if they accept the same language. The *size* of an NFA is the number of its transitions.

The *underlying digraph* of an NFA $A = (S, \Sigma, \delta, s_0, F)$ is the digraph $D = (S, E)$ such that $E = \{(s, s') \mid s, s' \in S \text{ and } \exists a \in \Sigma \cup \{\epsilon\} \text{ such that } (s, a, s') \in \delta\}$. Note that even there can be more than one symbol of $\Sigma$ between two states $s$ and $s'$, only one arc exists in the underlying graph. We call *initial vertex* the vertex that corresponds to the initial state, *final vertices* the ones that correspond to final states and *intermediate vertices*, all the others. An automaton is *useful* if in its underlying digraph, every vertex is in a path from the initial vertex to a final vertex. An automaton is *acyclic* if its underlying digraph is acyclic. We will use the above terminology both for digraphs and for automata.

An *extended finite automaton* (EFA) $A$ is a quintuple $(S, \Sigma, \delta, s_0, F)$, where $S$, $\Sigma$, $s_0$ and $F$ are as before and $\delta : S \times S \to R_\Sigma$. We assume that $\delta(s, s') = \emptyset$, if the transition from $s$ to $s'$ is not defined. A string $x \in \Sigma^\star$ is said to be accepted by $A$ if $x = x_1 \cdots x_n$, for $x_1, \ldots, x_n \in \Sigma^\star$ and there is a state sequence $s_0, s_1, \ldots, s_n$ with $s_n \in F$, such that $x_1 \in L(\delta(s_0, s_1))$,..., $x_n \in L(\delta(s_{n-1}, s_n))$. The language accepted by $A$ is the set of all strings accepted by $A$. The *size* of an EFA is the number of its transitions. The *underlying digraph* of an EFA is a digraph $D = (S, E)$ such that $(s, s') \in E$ if and only if $\delta(s, s') \neq \emptyset$. Any NFA can be easily transformed into an equivalent EFA, with the same underlying digraph: for each pair of states $(s, s')$ one needs to construct a regular expression $a_1 + \cdots + a_n$ such that $(s, a_i, s') \in \delta$, $a_i \in \Sigma \cup \{\epsilon\}$, $1 \leq i \leq n$.

Finally, we recall the conversion of an EFA $A$ into a regular expression $r$, using the *state elimination algorithm* (SEA). In each step, a non-initial and non-final state of the EFA is deleted and the transitions are changed in such way that the new EFA is equivalent to the older one. Formally, let $A = (S, \Sigma, \delta, s_0, F)$ be an EFA. Then

1. (a) If $s_0 \in F$ or exists $s \in S$ such that $\delta(s, s_0) \neq \emptyset$, then add a new state $\alpha$ to $S$, define $\delta(\alpha, s_0) = \epsilon$ and $\alpha$ is the new initial state.

   (b) If $|F| > 1$, then add a new state $\omega$ and transition $\delta(s, \omega) = \epsilon$, for all $s \in F$. The set of final states becomes $\{\omega\}$.

   Without lost of generality, let $A' = (S', \Sigma, \delta', \alpha, \{\omega\})$ denote the new EFA. We denote by $r_{ss'}$ the regular expression $\delta(s, s')$.

2. If $S' = \{\alpha, \omega\}$, then the resulting regular expression is $r_{\alpha\omega} r_{\omega\omega}^\star$, and the algorithm terminates. Otherwise continue to step 3.

3. **Choose** $s \in S' \setminus \{\alpha, \omega\}$. Eliminate $s$ from $A'$, considering $S' \setminus \{s\}$ the new set of states, and for each $s_1, s_2 \in S' \setminus \{s\}$,

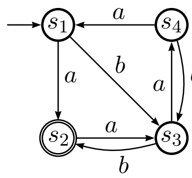
$$\delta'(s_1, s_2) = r_{s_1 s_2} + r_{s_1 s} r_{ss}^\star r_{s s_2},$$

Continue to step 2.

Let us observe that, in each step, if we have $s(k;l)$, the contribution of $s$ for the size of the final regular expression can be measured by

$$(k-1)\sum_{i=1}^{k}|r_{s_is}| + (l-1)\sum_{j=1}^{l}|r_{ss_j}| + (kl-1)|r_{ss}|. \tag{1}$$

This contribution is 0 if $s(1;1)$. To illustrate, this dependence from the order of state elimination in the resulting regular expression, consider the following automaton:




If the order of state elimination is $s_2$, $s_1$, $s_4$, and $s_3$ we obtain the regular expression $a+(b+aa)(ba+a(b+a(b+aa))^\star(b+aaa)$ with 16 alphabetic symbols. If the order is $s_3$, $s_4$, $s_2$, and $s_1$, the resulting expression is $ba(ba)^\star a+(a+bb+ba(ba)^\star bb)(ab+aa(ba)^\star bb)^\star aa(ba)^\star a)^\star(a+bb+ba(ba)^\star bb)(ab+aa(ba)^\star bb)^\star$ with $44$ alphabetic symbols. In each step, we could try to simplify the regular expression obtained, but our goal is to try to discover a state order that leads to shorter regular expressions. One of the advantage of this approach is to avoid the generation of bigger intermediate regular expressions.

## 3. SP Automata

In this section, we will consider only useful acyclic automata with one final state. The underlying digraphs of these automata are called in the literature (acyclic) *st*-digraphs [1]. In an *st*-digraph there exists only one vertex $\alpha$ of indegree 0 ($\alpha(0;m)$, for $m > 1$), only one vertex $\omega$ of outdegree 0 ($\omega(n;0)$, for $n > 1$), and each vertex occurs in some path from $\alpha$ to $\omega$. In an acyclic automaton, the vertices $\alpha$ and $\omega$ correspond to the initial and final state, respectively.

We are going to characterise a class of automata using the notion of digraph homeomorphism. Two digraphs are homeomorphic if both can be obtained from the same digraph by a sequence of subdivisions of arcs [16, 1]. Consider the digraph[1]

$$\vec{\mathsf{R}} = (\{s_1,s_2,s_3,s_4\},\{(s_1,s_2),(s_1,s_3),(s_2,s_3),(s_2,s_4),(s_3,s_4)\}),$$

represented in Figure 1.

**Definition 3.1.** A useful acyclic NFA with one final state is *SP* (series-parallel) if its underlying digraph does not contain a subgraph homeomorphic to $\vec{\mathsf{R}}$. We say that the underlying digraph is an *SP digraph*.

The digraph in Figure 2, is not SP, because its underlying digraph contains a subgraph homeomorphic to $\vec{\mathsf{R}}$, namely, the one obtained by excluding the vertex $s_3$ and the arcs $(s_4,s_6)$, and $(s_5,s_{10})$ (with dashed lines, in the figure). On the other hand, the digraph in Figure 3 is SP. The automaton presented in the introduction is obviously not SP.

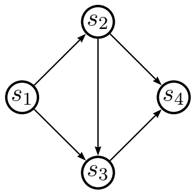



Figure 1. Digraph $\vec{\mathsf{R}}$.

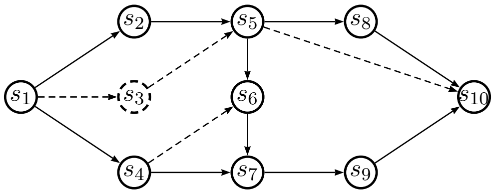



Figure 2. A non-SP digraph.

Series-parallel graphs and digraphs are extensively studied in the literature. Initially they were studied in relation with series-parallel electrical circuits by Riordan and Shannon [28]. Duffin [5] characterised SP graphs by a forbidden undirected subgraph characterisation as the one in Definition 3.1. This class of (di)graphs is also important because many NP-complete graph problems become polynomial (or even linear) when restricted to it [32, 34].

In our preliminary work [26] this class of automata was called UDR as an acronym of *Unique for the Distributivity Rule*. If an automaton is not SP, there are at least two states (with outdegree or indegree greater than 1) such that the order chosen to eliminate them leads to two different regular expressions, where one results from the application of a distributivity rule to the other. In general, one of the choices will lead to a shorter expression, but it is not easy to determine which. In the next section, we show that this is not the case if the automaton is SP. In an SP automaton, in each step there is always a state $s(1;1)$ that we can select to eliminate.

**Proposition 3.1.** Any acyclic NFA is equivalent to an SP automaton.

**Proof:**
Every acyclic NFA $A = (S, \Sigma, s_0, F)$ is equivalent to a regular expression in *disjunctive normal form, i.e.*, $r_1 + \cdots + r_n$ where for $1 \leq l \leq n$,

$$r_l = \begin{cases} \epsilon & \text{or,} \\ a_{l_1} \cdots a_{l_{k_l}} & a_{l_j} \in \Sigma, 1 \leq j \leq k_l. \end{cases}$$

From this r.e. we can construct an equivalent NFA

$$A' = (\{\alpha\} \cup \{\omega\} \cup \bigcup_{1 \leq l \leq n} S_l, \Sigma, \alpha, \{\omega\})$$

[1] $\vec{\mathsf{R}}$ is a directed version of the complete graph with 4 vertices ($\mathsf{K}_4$).

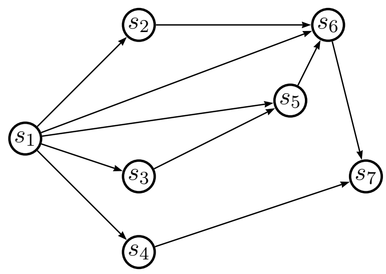



Figure 3. An SP digraph.

where $S_l = \{s_{l_1}, \dots, s_{l_{k_l}}\}$, $1 \leq l \leq n$, and such that

$$\begin{aligned} \delta'(\alpha, a_{l_1}) &= \{s_{l_1}\}, \\ \delta'(s_{l_j}, a_{l_{j+1}}) &= \{s_{l_{j+1}}\}, \quad \text{for } 2 \leq j \leq k_{l-1}, \\ \delta'(s_{l_{k_l}}, \epsilon) &= \{\omega\}. \end{aligned}$$

The automaton $A'$ is trivially SP. □

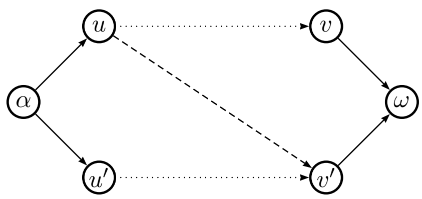
An interesting open problem is to design a method to obtain, from an acyclic NFA, an SP automaton with a minimal number of transitions (size).

Without loss of generality, we can just consider SP digraphs (instead of automata), $D = (S, E, \alpha, \omega)$, where $\alpha$ denotes the initial vertex and $\omega$ denotes the final vertex. It is obvious that,

**Lemma 3.1.** An $st$-digraph subgraph of an SP digraph is an SP digraph.

**Lemma 3.2.** Let $D = (S, E, \alpha, \omega)$ be an SP digraph. Let $\alpha(0; k)$, $\omega(m; 0)$, and $k, m > 1$. Suppose that $u$ and $u'$ are two distinct vertices adjacent from $\alpha$, and $v$ and $v'$ are two distinct vertices adjacent to $\omega$. If there are two disjoint paths $\overline{\alpha u \cdots v \omega}$ and $\overline{\alpha u' \cdots v' \omega}$ then there cannot exist a path $\overline{\alpha u \cdots v' \omega}$ nor a path $\overline{\alpha u' \cdots v \omega}$.

**Proof:**
Suppose that there exists a path $\overline{\alpha u \cdots v' \omega}$, partially in dashed line in the picture below:




It is obvious that there will be a subgraph of $D$ homeomorphic to $\vec{\mathsf{R}}$. This subgraph will contain the vertices $\{\alpha, u, s', \omega\}$ (corresponding to the vertices of $\vec{\mathsf{R}}$), where $s'$ is the first vertex common to the path $\overline{\alpha u \cdots v' \omega}$ and to the path $\overline{\alpha u' \cdots v' \omega}$ (at least $v'$ or $u'$). The proof is analogous if there exists a path $\overline{\alpha u' \cdots v \omega}$. □

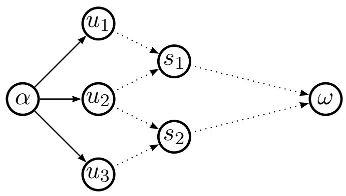
If $\alpha(0;k)$ and $k>1$, let $U$ be the set of the vertices adjacent from $\alpha$. For $u \in U$, let

$$X_u = \{u\} \cup \{s \mid s \in S \setminus \{\omega\} \text{ and there exists a path } \overline{u \cdots s}\}.$$

Consider the binary relation $\downarrow$ in $U \times U$, such that $u \downarrow u'$ if and only if $X_u \cap X_{u'} \neq \emptyset$.

**Lemma 3.3.** If $D$ is an SP digraph, the relation $\downarrow$ is an equivalence relation on $U$.

**Proof:**
The reflexivity and the symmetry are trivial, we only need to prove the transitivity. Let $u_1, u_2, u_3 \in U$, $u_1 \downarrow u_2$, and $u_2 \downarrow u_3$. Then there exist $s_1 \in X_{u_1} \cap X_{u_2}$ and $s_2 \in X_{u_2} \cap X_{u_3}$. If $s_1 = s_2$ then $u_1 \downarrow u_3$. Suppose that $s_1 \neq s_2$ and that there is no path $\overline{s_1 \cdots s_2}$ nor $\overline{s_2 \cdots s_1}$. Then, we will have the following diagram:




The vertices $\{\alpha, u_2, s_1, \omega\}$ define a subgraph homeomorphic to $\vec{\mathsf{R}}$, which contradicts the fact that $D$ is SP. □

**Lemma 3.4.** Let $D = (S, E, \alpha, \omega)$ be an SP digraph. Let $\alpha(0;k)$, $\omega(0;m)$, and $k, m > 1$. Let $U$ and $X_u$, for $u \in U$ be as above. If $[u]$ is an equivalence class of the relation $\downarrow$ with more than one element, then there exists $s \in S \setminus \{\omega\}$ such that:

1. $s \in \cap_{u' \in [u]} X_{u'}$
2. For all $s' \in \cup_{u' \in [u]} X_{u'}$ , every path $\overline{\alpha \cdots s'\omega}$ contains $s$.

**Proof:**
The existence of $s$, satisfying 1 is a direct consequence of the argument given in the proof of the transitivity of $\downarrow$. If condition 2 does not hold, then the digraph $D$ will have a subgraph homeomorphic to $\vec{\mathsf{R}}$. □

Now we can characterise the essential property of an SP digraph.

**Theorem 3.1.** Let $D = (S, E, \alpha, \omega)$ be an SP digraph and $|S| > 2$. Then $D$ has at least a vertex $s$ such that $s(1;1)$.

**Proof:**
The proof is made by induction on the number of vertices of the digraph, $|S| = n$. If $n = 3$, it is trivially true. The SP digraphs with $4$ vertices are enumerated in Figure 4, and it is easy to see that all of them have one vertex $s$ such that $s(1;1)$. Assume that the theorem holds for SP digraphs with less than $n > 4$ vertices. We want to show that the same is true for a $D = (S, E, \alpha, \omega)$ with $|S| = n$. If $\alpha(0;1)$, let

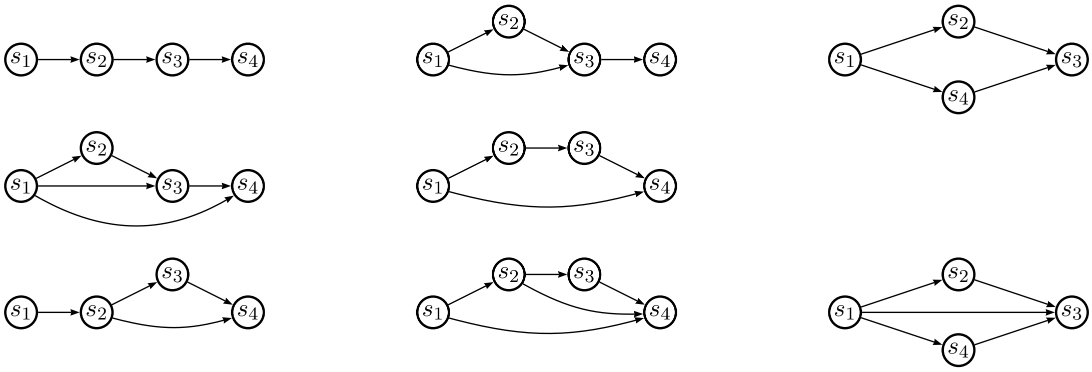



Figure 4. SP digraphs with 4 vertices.

$u \in S \setminus \{\omega\}$ be the vertex adjacent from $\alpha$. Then the digraph $D' = (S \setminus \{\alpha\}, E \setminus \{(\alpha, u)\}, u, \omega)$ is an SP digraph with $n-1$ vertices, and by induction hypothesis it has a vertex $s$ such that $s(1;1)$. An analogous argument can be given if $\omega(1;0)$. Let us suppose that $\alpha(0;k_\alpha)$ and $\omega(k_\omega;0)$, with $k_\alpha, k_\omega > 1$. As we are seeking for an intermediate vertex, we can ignore the arc $(\alpha, \omega) \in E$, if it exists. Let $U$, $X_u$ and $\downarrow$ be as defined above. For every $u \in U$, we have one of the following cases:

1. The class $[u]$ has an unique element. Let $D' = (X_u \cap \{\omega\}, E', u, \omega)$, where $E'$ has all the arcs of $D$ with vertices in $X_u \cap \{\omega\}$. Then by Lemma 3.1, $D'$ is an SP digraph and has less than $n$ vertices. If $|X_u \cap \{\omega\}| = 2$, then $u(1;1)$ in $D$. Otherwise, by induction hypothesis, $D'$ has a vertex $s'$ such that $s'(1;1)$, and, in $D$ we have also $s'(1;1)$ (by Lemma 3.2).

2. The class $[u]$ has more than one element. Let $D' = (\cap_{u' \in [u]} X_{u'}, E', \alpha, s)$, where $E'$ has all the arcs of $D$ with vertices in $\cup_{u' \in [u]} X_{u'}$ and $s$ as defined in Lemma 3.4. Then by Lemma 3.4, $D'$ is an SP digraph and has less than $n$ vertices. By induction hypothesis, it has a vertex $s'$ such that $s'(1;1)$, and, in $D$ we have also $s'(1;1)$.

$\square$

**Theorem 3.2.** Let $A = (S, \Sigma, \alpha, \delta, \omega)$ be a useful acyclic NFA with an unique final state. We can obtain a regular expression equivalent to $A$ using the state elimination algorithm (SEA) in such way that in each step we remove a state $s$ with $s(1;1)$ if and only if $A$ is SP.

**Proof:**
Suppose that $A$ is SP. If $S = \{\alpha, \omega\}$ then there is nothing to be proven. Otherwise, by Theorem 3.1, we can choose a state $s$ to eliminate such that $s(1;1)$. The resulting EFA $A'$ is SP, since with that elimination step no automaton state increases its indegree nor its outdegree (at most the adjacent state to $s$ decreases its outdegree by one and the adjacent state from $s$ decreases its indegree by one).

If we apply the SEA to a useful acyclic non-SP EFA $A$, then the underlying digraph of $A$ has a subgraph homeomorphic to $\vec{\mathsf{R}}$. Let $s_1$, $s_2$, $s_3$, and $s_4$ be the corresponding vertices. All of them have either indegree or outdegree greater than 1, and all those degrees cannot decrease to 1 unless one of the states is eliminated. $\square$

**Corollary 3.1.** Given an SP automaton $A$, it is possible to construct an equivalent regular expression with size linear in the size of $A$.

**Proof:**
In the application of the SEA, in each step if $s$ is the state to remove (and $s(1;1)$), then there exists one and only one pair of states $(s_1, s_2)$ such that $\delta'(s_1, s_2) = r_{s_1 s_2} + r_{s_1 s} r_{s s_2}$ and all the other transitions are not changed. The size of the regular expression obtained is the number of transitions of $A$ with alphabetic symbols (counting its multiplicities). □

## 4. An algorithm to decide if a digraph is SP

Fortune *et al.* [9] have shown that the problem of determining if an acyclic digraph $D = (V, E)$ has a subgraph homeomorphic to a fixed digraph $P = (V', E')$ has a polynomial time algorithm $\mathcal{O}(n^{k+s})$, where $n = |V|$, $k = |E'|$ and $s = |V'|$. However, to determine if a digraph is SP, specialised algorithms can be designed. Recognition of series-parallels graphs can be done in linear time [34] and several parallel algorithms were described in the 1990's [17, 7, 3]. We present a new sequential algorithm that can be easily adapted to obtain a short regular expression.

Let us suppose that we have already determined that $D = (V, E, \alpha, \omega)$ is an (acyclic) *st*-digraph, with a topological ordering $o$ (this can be achieved in $\mathcal{O}(n + m)$). For $v \in V$, let $Adj(v)$ be a list of vertices adjacent to $v$, let $od(v)$ be the outdegree of $v$, let $id(v)$ be the indegree of $v$ and let $Inc(v)$ be a list of vertices adjacent from (incident to) $v$.

In Figure 5 we present the algorithm in pseudo-code. The vertices of the digraph are going to be traversed in topological order. Each arc $(u, v) \in E$ is annotated with a list of relevant vertices with outdegree greater than 1 that precedes $v$ (in a path from $\alpha$). Those labels are denoted by $A(u, v)$ and they can be a list of vertices or a reference to another equivalent annotation. We write $a \equiv b$ to denote that $a$ becomes a reference to $b$ and we write $a \not\equiv b$ to test if $a$ is not a reference to $b$. We use $\leftarrow$ as the standard assignment operator. The empty list is denoted by [] and $l.v$ represents the concatenation of $v$ with the list $l$. We use other standard list operations as $first$ (first element of the list), $last$ (last element of the list) and $butlast$ (the list without the last element).

The algorithm proceeds as follows. While the visited vertices have indegree less than 2, *i.e.* they are not a confluence, the relevant predecessors are collected (lines 17–21). If a vertex $v$ has indegree greater than 1, then either we can "resolve" all the precedent bifurcations or the digraph must be non SP (lines 4–16). A confluence is "resolved" if there are two vertices, $w$ and $u$ adjacent to $v$ with labels that have equal values but which are not references to the same object. In this case, those labels can be unified, and if $w'$ is the last element of those labels, $od(w')$ is decreased by 1. When that value is 1, the vertex $u$ is no longer relevant and can be deleted from the labels (lines 8–15).

**Example 4.1.** Consider the digraph in Figure 6, which has the vertices' labels topologically ordered. The algorithm execution can be summarised as follows:

1. For $s_0$, as $od(s_0) = 1$ we have $A(s_0, s_1) \leftarrow [\,]$.
2. For $s_1$, as $od(s_1) = 2$ we have $A(s_1, s_2) \leftarrow [s_1]$ and $A(s_1, s_8) \leftarrow [s_1]$.
3. For $s_2$, as $od(s_2) = 2$ we have $A(s_2, s_3) \leftarrow [s_1, s_2]$ and $A(s_2, s_8) \leftarrow [s_1, s_2]$.

```
def spgp(S,E):
  for v ∈ S (S topologically ordered)
    e ← |Inc(v)|
    while e > 1 do
      m ← max{|A(u,v)| | u ∈ Inc(v)}
      L ← [u ∈ Inc(v) | |A(u,v)| = m]
      w ← first(L)
      if (∃u ∈ L \ {w}) (A(u,v) ≢ A(w,v)) ∧ (A(u,v) = A(w,v)) then
        w' ← last(A(w,v))
        od(w') ← od(w') − 1
        if od(w') = 1 then
          A(u,v) ← butlast(A(u,v))
        A(w,v) ≡ A(u,w)
        Inc(v) ← Inc(v) \ {u}
        e ← e − 1
      else return 0
    if v = α then p ← [ ]
    else p ≡ A(first(Inc(v)),v)
    for v' ∈ Adj(v)
      if od(v) ≠ 1 then A(v,v') ← p.v
      else A(v,v') ≡ p
  return 1
```

Figure 5. Determining if a digraph is SP.

4. For $s_3$, as $od(s_3) = 3$ we have $A(s_3, s_4) \leftarrow [s_1, s_2, s_3]$, $A(s_3, s_6) \leftarrow [s_1, s_2, s_3]$, and $A(s_3, s_8) \leftarrow [s_1, s_2, s_3]$.

5. For $s_4$, as $od(s_4) = 2$ we have $A(s_4, s_5) \leftarrow [s_1, s_2, s_3, s_4]$ and $A(s_4, s_6) \leftarrow [s_1, s_2, s_3, s_4]$.

6. For $s_5$, as $od(s_5) = 1$ we have $A(s_5, s_6) \equiv A(s_4, s_5)$. The state $s_5$ is not relevant, the arc must be equivalent to its predecessor.

7. For $s_6$, as $id(s_6) = 3$ (is a confluence) and $A(s_4, s_6) = A(s_3, s_6)$ (one of the arcs can be resolved), we have $A(s_5, s_6) \equiv A(s_4, s_6)$, $od(s_4) \leftarrow od(s_4) - 1$, and $id(s_6) \leftarrow id(s_6) - 1$.

8. Because $od(s_4) = 1$, we have $A(s_4, s_6) \leftarrow butlast(A(s_4, s_6))$, *i.e.*, $A(s_4, s_6) \leftarrow [s_1, s_2, s_3]$. The state $s_4$ is no more relevant and so can be removed from the annotations.

9. As $id(s_6) = 2$ (the confluence is not yet resolved) and $A(s_3, s_6) = A(s_4, s_6)$ (there is one more arc to solve), then $A(s_3, s_6) \equiv A(s_4, s_6)$, $od(s_3) \leftarrow od(s_3) - 1$, and $id(s_6) \leftarrow id(s_6) - 1$.

10. As $od(s_3) = 1$ we have $A(s_3, s_6) \leftarrow butlast(A(s_3, s_6))$, *i.e.* $A(s_3, s_6) \leftarrow [s_1, s_2]$.

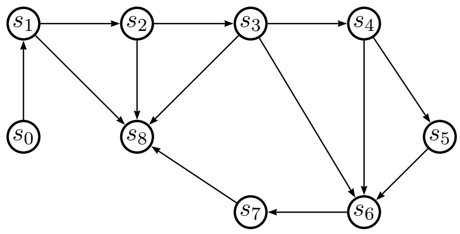



Figure 6. Example of an SP digraph.

11. The confluence in $s_6$ is already solved, because $id(s_6) = 1$. We can go on. As $od(s_6) = 1$ the vertex is not relevant and $A(s_6, s_7) \equiv A(s_5, s_6)$.

12. For $s_7$, as $od(s_7) = 1$, $A(s_7, s_8) \equiv A(s_6, s_7)$.

13. It remains to process $s_8$. As $id(s_8) = 3$ (it is a confluence) and $A(s_7, s_8) = A(s_2, s_8)$, we have $A(s_2, s_8) \equiv A(s_7, s_8)$, $od(s_2) \leftarrow od(s_2) - 1$, and $id(s_8) \leftarrow id(s_8) - 1$.

14. Now $od(s_2) = 1$, and $s_2$ can be removed from the annotations, $A(s_2, s_8) \leftarrow butlast(A(s_2, s_8))$ (*i.e.*$[s_1]$).

15. Although an arc was eliminated, the confluence remains in $s_8$: $id(s_8) = 2$. As $A(s_1, s_8) = A(s_7, s_8)$, we have $A(s_1, s_8) \equiv A(s_7, s_8)$, $od(s_1) \leftarrow od(s_1) - 1$, and $id(s_8) \leftarrow id(s_8) - 1$.

16. The vertex $s_1$ is no more relevant (because $od(s_1) = 1$), so $A(s_1, s_8) \leftarrow butlast(A(s_1, s_8))$ (*i.e.* $A(s_1, s_8) \leftarrow [\,]$), and all annotations have now the value $[\,]$.

17. As $id(s_8) = 0$ all confluences where resolved and the algorithm returns the value 1.

**Theorem 4.1.** The algorithm spgp is correct and has time complexity $\mathcal{O}(n^2 \log n)$.

**Proof:**
*(Sketch)* If the digraph $D$ is not SP, it has a subgraph homeomorphic to $\vec{\mathsf{R}}$. Let $s_1$, $s_2$, $s_3$, and $s_4$ be the corresponding vertices. The vertex $s_3$ has indegree greater than 1. When $s_3$ is visited, there are $v, v' \in Inc(s_3)$ such that $s_1 \in A(v, s_3)$ and $s_1, s_2 \in A(v', s_3)$. Those labels can never refer to the same object, thus the algorithm must return 0. Suppose that the algorithm returns 0. Then there exists $v \in V$ such that $v(e; k)$ with $e > 1$ and there exist $v', v'' \in Inc(v)$ such that $A(v', v)$ and $A(v'', v)$ are different. Let $u \in (A(v', v) \setminus A(v'', v))$. The outdegree of $u$ is greater than 1 (otherwise it was not relevant) and $u \neq \alpha$. Then there must exist two disjoint paths $\overline{\alpha \cdots u \cdots v}$ and $\overline{\alpha \cdots v}$. There exists a path $\overline{u \cdots \omega}$ that does not go through $v$ and a path $\overline{v \cdots \omega}$ that does not go through $u$. Thus $D$ has a subgraph homeomorphic to $\vec{\mathsf{R}}$, defined by $\alpha, u, v, s'$, where $s'$ is a common ancestor of $u$ and $v$ (at least $\omega$). Thus the digraph $D$ is not SP.

To analyse the time complexity of the algorithm, first note that the total cost of executing lines 17–21 is $\mathcal{O}(m)$. For each $v \in V$, the lines 4–16 are, in the worst case, executed in $\mathcal{O}(n \log n)$ (as the annotations can be sorted only once). Thus the time complexity of spgp is $\mathcal{O}(n^2 \log n)$. $\square$

**Proposition 4.1.** If $A$ is SP, the algorithm spgp can be used to compute an equivalent regular expression with size linear in the size of $A$.

**Proof:**
Let $A = (S, \Sigma, \delta, \alpha, \omega)$ be an EFA. In the algorithm spgp we can extend the arc annotations to contain the automaton's transition labels. For $(s, s') \in S$, let $A(s, s')$ be the list of vertices as before and let $R(s, s')$ be the associated regular expression. Initially, these regular expressions are disjunction of alphabetic symbols according to the transition relation $\delta$,

$$(\forall s, s' \in S) R(s, s') = \sum_{\delta(s,a)=s'} a$$

Whenever a state $s_1(k; 1)$ is visited and its incident arcs have been resolved (lines 17–21) let $rp$ be the regular expression correspondent to the label $p$. Note that in line 17, $p$ is $\epsilon$ and in line 26 its value must be $R(first(Inc(v), v))$. Then, in line 22, we add the instruction:

$$R(v, v') \leftarrow\ rp \cdot R(v, v').$$

Whenever a confluence is resolved (in line 21) we add the instruction:

$$R(u, v) \leftarrow\ R(u, v) + R(w, v).$$

And in line 12, we add another concatenation, if $w'$ is not the initial state:

$$R(w, v) \leftarrow\ rp \cdot R(w, v),$$

where $rp$ is the regular expression $R(first(Inc(w')), w')$ (there must be at most one). In the end we obtain the same regular expression of Corollary 3.1, as $R(\alpha, \omega)$. □

# 5. Counting SP digraphs

In the context of electrical networks Riordan and Shannon [28] presented a formula for counting series-parallel networks that arise from combinations of resistances in series and in parallel. The correspondent digraphs differ from the ones we consider here as they allow multiple edges between the same order pair of vertices (*multidigraphs)*. They give an enumerative formula (based on a generating function) for the number of (unlabelled) digraphs by number of edges and the correspondent sequence appears as number A000084 in the OEIS [31]. Many more related enumerative formulas appear in the OEIS, and were studied, for instance, by Moon [23], Golinelli [12] and Finch [8]. More recently Bodirsky *et al.* [2] studied the asymptotically behaviour for the number of labelled series-parallel graphs.

In this section, we present recursive formulae for the number of SP (unlabelled) digraphs by number of vertices. We begin by giving a definition of SP digraphs in terms of compositions of series and parallel operations. This definition is similar to the ones usually called two-terminal series-parallel networks (see [34]), but differs by not allowing multiple edges. With some assumed abuse of notation we refer this set as SP.

**Definition 5.1. (SP II)**
The set of series-parallel SP digraphs is defined recursively as follows:

(i) $\mathsf{L} = (\{\alpha, \omega\}, \{(\alpha, \omega)\}, \alpha, \omega) \in \mathsf{SP}$

(ii) If $D = (V, E, \alpha, \omega) \in \mathsf{SP}$ and $D' = (V', E', \alpha', \omega') \in \mathsf{SP}$ such that $V \cap V' = \emptyset$, then $\mathsf{S}DD' = (V'', E'', \alpha, \omega') \in \mathsf{SP}$, where

$$V'' = (V \cup V' \cup \{\beta\}) \setminus \{\omega, \alpha'\},$$

$$\begin{aligned} E'' = &\{(v, v') \in E \mid v' \neq \omega\} \ \cup \ \{(v, \beta) \mid (v, \omega) \in E\} \cup \\ \cup \ &\{(v, v') \in E' \mid v \neq \alpha'\} \ \cup \ \{(\beta, v') \mid (\alpha', v') \in E'\}. \end{aligned}$$

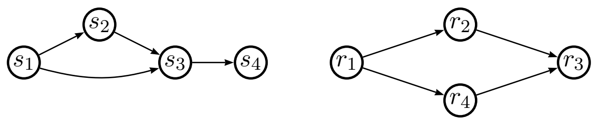
(iii) Let $D = (V, E, \alpha, \omega) \in \mathsf{SP}$ and $D' = (V', E', \alpha', \omega') \in \mathsf{SP}$ where $V \cap V' = \emptyset$, $(\alpha, \omega) \notin E$, and $(\alpha', \omega') \notin E'$, then $\mathsf{P}DD' = (E'', V'', \alpha'', \omega'') \in \mathsf{SP}$, where

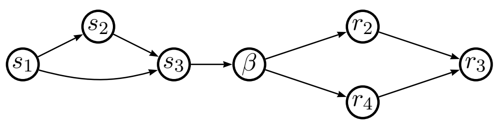
$$V'' = (V \cup V' \cup \{\alpha'', \omega''\}) \setminus \{\alpha, \omega, \alpha', \omega'\},$$

$$\begin{aligned} E'' = &\{(v, v') \in E \mid v \neq \alpha \ \wedge \ v' \neq \omega\} \ \cup \{(\alpha'', v') \mid (\alpha, v') \in E\} \cup \\ \cup \ &\{(v, \omega'') \mid (v, \omega) \in E\} \ \cup \{(v, v') \in E' \mid v \neq \alpha' \ \wedge \ v' \neq \omega'\} \cup \\ \cup \ &\{(\alpha'', v') \mid (\alpha', v') \in E'\} \ \cup \ \{(v, \omega'') \mid (v, \omega') \in E'\}. \end{aligned}$$

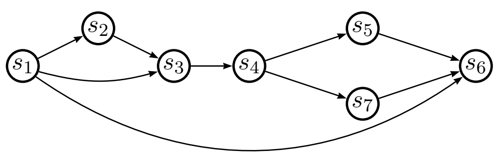
(iv) If $D = (V, E, \alpha, \omega) \in \mathsf{SP}$ and $(\alpha, \omega) \notin E$, then the digraph $\mathsf{P}D\mathsf{L} = (V, E \cup \{(\alpha, \omega)\}, \alpha, \omega) \in \mathsf{SP}$.

**Example 5.1.** Applying rule (ii) to the first two SP digraphs we obtain that the third digraph is SP:

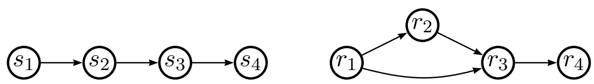


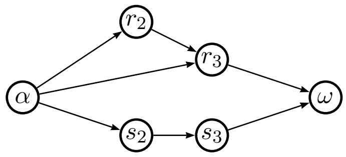




Applying the rule (iv) to the last digraph above we obtain that following digraph is SP:




In the same manner, and applying rule (iii) to the first two SP digraphs bellow, we conclude that the third digraph is SP:

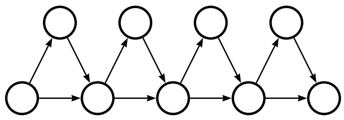




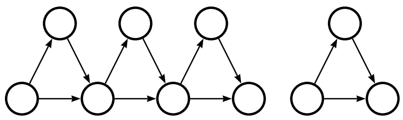

The following theorem is analogous to other in the literature [34] and, for completeness, we give a proof in Appendix 8.

**Theorem 5.1.** Definitions 5.1 and 3.1 define the same set of digraphs.

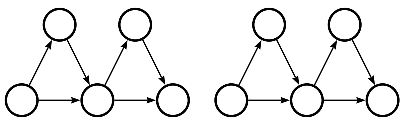
To count the number of SP digraphs with $n$ vertices we can use the Definition 5.1 with a modification of rule (ii) to avoid double counting some identical digraphs. We illustrate the problem with a simple example. Consider the following digraph:



It can be built using rule (ii) and considering the following two digraphs:



Or considering this other pair of digraphs:



So when applying rule (ii) we must only take digraphs that are *indecomposable* by rule (ii). Two digraphs are *indecomposable* by rule (ii) if the last rule used in their construction was not rule (ii).

Applying rule (ii) to two digraphs $D$ and $D'$ with, $n$ and $n'$ vertices respectively, a digraph $\mathsf{S}DD'$ with $n + n' - 1$ vertices is obtained. If we iterate this operation for digraphs $(D_i)_{i\in[1,k]}$ with $(n_i)_{i\in[1,k]}$ vertices, respectively, we obtain a digraph which number of vertices is

$$\left(\sum_{i\in[1,k]} n_i\right) - k + 1.$$

Let $U_n$ be the number of SP digraphs with $n$ vertices, and let $U_n^{(ii)}$ be the number of SP digraphs with $n$ vertices which last rule of construction was rule (ii). We have

$$U_n^{(ii)} = \sum_{\tau} \prod_{j\in[1,k]} (U_{n_j} - U_{n_j}^{(ii)})^{p_j} \frac{k!}{\prod_{l\in[1,k]} p_l!}, \tag{2}$$

where $\tau$ runs over all sets $\{(n_j, p_j)\}_{j\in[1,k]}$ with $k > 1$, $(\forall j \in [1,k])\,(n_j \geq 2 \ \wedge \ p_j > 0)$ and

$$\left(\sum_{j\in[1,k]} n_j p_j\right) - k + 1 = n.$$

Let $U_n^{(iii)}$ be the number of SP digraphs with $n$ vertices which last rule of construction was rule (iii). Because the parallel composition is commutative, when obtaining a digraph P$DD'$ with rule (iii), the number of vertices of $D$ is not smaller than the number of vertices of $D'$. Applying rule (iii) to two digraphs $D$ and $D'$ with, $n$ and $n'$ vertices respectively, a digraph P$DD'$ with $n + n' - 2$ vertices is obtained. The number of these digraphs is then given by the following formula:

$$U_n^{(iii)} = \sum_{\substack{n'+n''=n+2\\ n'\geq n''\\ n''>2}} (U_{n'}^{(ii)} + U_{n'}^{(iii)})(U_{n''}^{(ii)} + U_{n''}^{(iii)}). \tag{3}$$

It remains to determine the number of SP digraphs produced by rule (iv). These digraphs can be obtained from any SP digraph with $n$ vertices, produced by rules (ii) or (iii). So its number is

$$U_n^{(iv)} = U_n^{(ii)} + U_n^{(iii)}. \tag{4}$$

The total number of SP digraphs with $n$ vertices is

$$U_n = U_n^{(ii)} + U_n^{(iii)} + U_n^{(iv)}. \tag{5}$$

Now we can simplify equation (3) to

$$U_n^{(iii)} = \sum_{\substack{n'+n''=n+2\\ n'\geq n''\\ n''>2}} \frac{U_{n'} U_{n''}}{4}. \tag{6}$$

To complete the counting we consider the following base values:

$$U_2 = 1, \quad U_3 = 2, \quad U_3^{(ii)} = 1, \quad U_3^{(iii)} = 0, \quad U_3^{(iv)} = 1.$$

Table 1 summarises the number of SP digraphs with no more than 20 vertices.

This enumeration process was easily adapted for a generation algorithm of SP digraphs, and so of SP automata.

Table 1. Number of SP digraphs for some small values of $n$.

| $n$ | | $n$ | |
|---|---|---|---|
| 2 | 1 | 3 | 2 |
| 4 | 8 | 5 | 38 |
| 6 | 228 | 7 | 1382 |
| 8 | 9342 | 9 | 62944 |
| 10 | 453724 | 11 | 3235216 |
| 12 | 24131728 | 13 | 178448548 |
| 14 | 1364523112 | 15 | 10339603930 |
| 16 | 80365190044 | 17 | 620051361254 |
| 18 | 4883464795602 | 19 | 38186977218324 |
| 20 | 303643041719194 | | |

## 6. Other related work

There are not many papers in the literature on the characterisation of the conversion from NFA's to regular expressions, as was pointed by Ellul *et al.* [6]. If the automaton's underlying graph is planar it is possible to obtain an equivalent regular expression of size less than $e^{\mathcal{O}(\sqrt{(n)})}$, where $n$ is the number of states of the automaton [6]. Generalising this result and based on graph separator techniques, Gruber and Holzer [13] presented algorithms that when applied to $n$-state deterministic finite automata obtain regular expressions of size $\mathcal{O}(2.6^n)$.

Giammarresi *et al.* [11] characterised the automata generated by the Thompson method [33] for converting regular expressions to automata. They called the underlying digraphs, Thompson digraphs. By induction on the structure of those digraphs we can prove that

**Corollary 6.1.** Every acyclic Thompson digraph is SP.

In the same way, Caron and Ziadi [4] characterised the automata generated by the Glushkov method for transforming regular expressions into finite automata [35]. As these automata may have more than one final state, we cannot directly compare the acyclic Glushkov digraphs and the SP digraphs. Introducing $\epsilon$-transitions it is possible to obtain an SP automata equivalent to an acyclic Glushkov automaton. It is also easy to see that every SP automaton is an acyclic Glushkov automaton.

More recently Han and Wood [15] studied the relation between the size of the resulting regular expression and the state elimination order in the SEA algorithm. They proved that in order to obtain a more succinct expression, states that correspond to cutvertices (on the underlying digraph) should be eliminated after all others. Gulan and Fernau [14], integrated the method presented in this paper and in Morais *et al.* [25, 26] into an iterative heuristic for general NFA.

Sakarovitch [29, 30] analysed the conversion algorithms between equivalent regular expressions and finite automata, and establishes some relationships between the regular expressions obtained from a given automaton by applying different methods of computing a regular expression from an automaton.

Lombardy and Sakarovitch [22] consider the problem of retrieving an automaton $\mathcal{A}$ from an equivalent regular expression that was obtained from $\mathcal{A}$. Although their main focus is not on the size of the representations, their approach clarifies the relationships between these conversions and can lead to better characterisations. Recently [21] they corrected the proof of their main theorem that states that if an NFA is co-deterministic it is possible to retrieve a new co-deterministic NFA.

## 7. Conclusion

In this work we show that if the underlying digraph of an acyclic NFA is series-parallel then it is possible to obtain in time $\mathcal{O}(n^2 \log n)$ an equivalent regular expression of size $\mathcal{O}(n)$. We also presented an enumerative formula for the number of $\mathsf{SP}$ digraphs with $n$ vertices, $U_n$. For now, we do not know a (closed form) generating function for $U_n$, and so no asymptotic analysis was possible. We plan to explore the methodology purposed by Fusy *et al.* [10] for the enumeration of families of graphs.

## 8. Acknowledgements

J.J. Morais contributed for the first characterisation of $\mathsf{SP}$ automata in terms of its underlying digraph exclude subgraphs homeomorphic to $\vec{\mathsf{R}}$ and developed an $\mathcal{O}(n^5)$ algorithm for determining if an acyclic NFA has that characteristic [24].

## A. Appendix

**Theorem A.1.** Definitions 5.1 and 3.1 define the same set of digraphs.

**Proof:**
First we show by induction on the number of rules' applications that a digraph described by Definition 5.1 does not contain a subgraph homeomorphic to $\vec{\mathsf{R}}$ (Definition 3.1). This is true if the digraph is $\mathsf{L}$. Suppose that all digraphs obtained by less than $n$ applications of rules (i)-(iv) do not contain a subgraph homeomorphic to $\vec{\mathsf{R}}$. Let $D = (V, E, \alpha, \omega)$ be a digraph that results from $n$ applications of rules (i)-(iv). The last rule must be one of rules (ii)-(iv). In the case of rule (ii), $D = \mathsf{S}D'D''$ and $D$ is obtained by the serial composition of two digraphs $D' = (V', E', \alpha', \omega')$ and $D'' = (V'', E'', \alpha'', \omega'')$, that by induction hypothesis do not contain a subgraph homeomorphic to $\vec{\mathsf{R}}$. Any path $\overline{v' \cdots v''}$ (in D), with $v' \in V'$ and $v'' \in V''$ must contain $\beta$, so if $D$ contained a subgraph homeomorphic to $\vec{\mathsf{R}}$ it should be a subgraph of $D'$ or $D$ , which is an absurd. If the last rule applied was rule (iii), then $D = \mathsf{P}D'D''$. In this case, there are no paths between any vertex of $D'$ and any vertex of $D''$, different from $\alpha$ and $\omega$. So the conclusion follows, as in the previous case. Finally, suppose that the last rule is rule (iv) and $D = \mathsf{P}D'\mathsf{L}$. The arc $(\alpha, \omega) \notin E'$ and it is easy to see that if $D$ contained a subgraph homeomorphic to $\vec{\mathsf{R}}$ it should be a subgraph of $D'$, which is an absurd.

We show now by induction on the number of arcs that if a digraph does not contain a subgraph homeomorphic to $\vec{\mathsf{R}}$, then it can be built by rules (i)-(iv). It is easy to see that all digraphs with equal or less than 4 arcs that has that property can be built using rules (i)-(iv) (see Figure 4, for $n = 4$). Suppose that the statement is valid for all digraphs with less that $n$ arcs and that satisfy Definition 3.1. Let $D = (V, E, \alpha, \omega)$ be a digraph with $n$ arcs and that does not contain a subgraph homeomorphic to $\vec{\mathsf{R}}$. If $D$ has a cutvertex $\beta$, that is a vertex that belongs to all paths $\overline{\alpha\cdots\omega}$, then $D = \mathsf{S}D'D''$ with $D' = (V', E', \alpha, \beta)$ and $D'' = (V'', E'', \beta, \omega)$. Every vertex in $D$ belongs to some path $\overline{\alpha\cdots\omega}$, so we take $V'$ as being the set of all vertices that belong to a path $\overline{\alpha\cdots\beta}$ and $E'$ as being the natural restriction of $E$ to $V'$. The digraph $D''$ is defined in the same way. $D$ and $D''$ can not contain a subgraph homeomorphic to $\vec{\mathsf{R}}$ and by induction hypothesis they can be built using rules (i)-(iv). Thus $D$ verifies the Definition 5.1. If $D$ does not have a cutvertex then there exist two paths $A$ and $B$ from $\alpha$ to $\omega$ that do not intersect each other (see for instance [16, page 27]). If $A$ or $B$ is an arc, then $D' = (V, E\backslash\{(\alpha,\omega)\}, \alpha, \omega)$ does not contain a subgraph homeomorphic to $\vec{\mathsf{R}}$ and $D = \mathsf{P}D'\mathsf{L}$. Thus $D$ verifies the Definition 5.1. If neither $A$ nor $B$ is an arc, then, as in Lemma 3.2, there can not exist a path between any vertex in $A$ and any vertex of $B$, different from $\alpha$ and $\omega$. Let $V'$ be the set of vertices of $D$ belonging to paths $\overline{\alpha\cdots\omega}$ with non empty intersection with $A$. Let $D' = (V', E', \alpha, \omega)$, where $E'$ is the natural restriction of $E$ to $V'$. Let $D'' = (V \setminus V', E'', \alpha, \omega)$ be defined analogously by considering the path $B$. Then, $D'$ and $D''$ can not contain a subgraph homeomorphic to $\vec{\mathsf{R}}$ and by induction hypothesis they can be built using rules (i)-(iv). But then $D = \mathsf{P}D'D''$ results from applying rule (iii) to $D'$ and $D''$. $\square$

# References


[1] Bang-Jensen, J., Gutin, G.: *Digraphs: Theory, Algorithms, and Applications*, Monographs in Mathematics, Springer-Verlag, 2001.

[2] Bodirsky, M., Gimenez, O., Kang, M., Noy, M.: Enumeration and limit laws of series-parallel graphs, *European Journal on Combinatorics*, **28**(8), November 2007, 2091–2105.

[3] Bodlaender, H. L., de Fluiter, B.: Parallel Algorithms for Series Parallel Graphs, *European Symp. on Algorithms*, 1996.

[4] Caron, P., Ziadi, D.: Characterization of Glushkov automata, *Theor. Comput. Sci.*, **233**(1-2), 2000, 75–90.

[5] Duffin, R. J.: Topology of Series-Parallel Networks, *J. of Math. Analysis and Appl.*, **10**, 1965, 303–318.

[6] Ellul, K., Krawetz, B., Shallit, J., Wang, M.: Regular Expressions: New Results and Open Problems, *Journal of Automata, Languages and Combinatorics*, **10**(4), 2005, 407–437.

[7] Eppstein, D.: Parallel Recognition of Series-Parallel Graphs, *Inf. Comput.*, **98**(1), 1992, 41–55.

[8] Finch, S.: *Mathematical Constants*, chapter Series-parallel networks, Cambridge University Press, 2003.

[9] Fortune, S., Hopcroft, J., Wyllie, J.: The directed subgraph homeomorphism problem, *Theoretical Computer Science*, **10**, 1980, 111–121.

[10] Fusy, E., Kang, M., Shoilekova, B.: A Complete Grammar for Decomposing a Family of Graphs into 3-connected Components, Submitted, Nov. 2007.

[11] Giammarresi, D., Ponty, J.-L., Wood, D., Ziadi, D.: A characterization of Thompson digraphs, *Discrete Applied Mathematics*, **134**(1-3), 2004, 317–337.

[12] Golinelli, O.: *Asymptotic behavior of two-terminal series-parallel networks*, Technical Report t97/074, CEA Sacley, Servive de Physique Théorique, 1997.

[13] Gruber, H., Holzer, M.: Provably Shorter Regular Expressions from Deterministic Finite Automata, *Proceedings of the 12th International Conference Developments in Language Theory* (M. Ito, M. Toyama, Eds.), number 5257, Springer, Kyoto, September 2008.

[14] Gulan, S., Fernau, H.: Local elimination-strategies in automata for shorter regular expressions, *SOFSEM 2008, Nový Smokovec, Slovakia, 2008, Volume II - Student Research Forum* (V. Geffert, J. Karhumäki, A. Bertoni, B. Preneel, P. Návrat, M. Bieliková, Eds.), 2008, ISBN 978-80-7097-697-5.

[15] Han, Y.-S., Wood, D.: Obtaining Shorter Regular Expressions from Finite-State Automata, *Theor. Comp. Science*, **370**, 2007, 110–120.

[16] Harary, F.: *Graph Theory*, 6th edition, Addison Wesley, 1969.

[17] He, X., Yesha, Y.: Parallel Recognitions and Decomposition of Two Terminal Series Parallel Graphs, *Inf. Comput.*, **75**(1), 1987, 15–38.

[18] Hopcroft, J., Motwani, R., Ullman, J. D.: *Introduction to Automata Theory, Languages and Computation*, Addison Wesley, 2000.

[19] Jiang, T., Ravikumar, B.: Minimal NFA problems are hard, *SIAM Journal of Computation*, 1993, 1117–1141.

[20] Kleene, S. C.: Representation of events in nerve nets and finite automata, in: *Automata Studies* (C. E. Shannon, J. McCarthy, Eds.), Princeton University Press, 1956, 3–41.

[21] Lombardy, S., Sakarovitch, J.: Corrigendum to our paper: How expressions can code for automata, Http://www.infres.enst.fr/ jsaka/PUB/pub.html.

[22] Lombardy, S., Sakarovitch, J.: How expressions can code for automata, *RAIRO- Theoret. Informatics and Appl.*, **39**, 2005, 217–237.

[23] Moon, J. W.: Some enumerative results on series-parallel networks, *Annals Discrete Math.*, 1987, 199–226.

[24] Morais, J. J.: *Computing small regular expressions from finite automata.*, Master Thesis, DCC-FCUP, 2005.

[25] Morais, J. J., Moreira, N., Reis, R.: *Acyclic Automata with easy-to-find short regular expressions*, Technical Report DCC-2005-03, DCC-FC& LIACC, Universidade do Porto, April 2005.

[26] Morais, J. J., Moreira, N., Reis, R.: Acyclic Automata with easy-to-find short regular expressions, *CIAA 2005, Tenth International Conference on Implementation and Application of Automata* (I. L. Jacques Farré, S. Schmitz, Eds.), 3845, Springer-Verlag, 2005.

[27] Reis, R.: *Autómatos:manipulação, geração e enumeração*, Ph.D. Thesis, Faculdade de Ciências da Universidade do Porto, 2007.

[28] Riordan, J., Shannon, C. E.: The number of two terminal series parallel networks, *J. Mathmatical Physics*, **21**(83-93), 1942.

[29] Sakarovitch, J.: *Élements de théorie des automates*, Vuibert Informatique, 2003.

[30] Sakarovitch, J.: The Language, the Expression, and the (Small) Automaton., *Implementation and Application of Automata, 10th International Conference, CIAA 2005*, 3845, Springer, 2006.

[31] Sloane, N.: The On-line Encyclopedia of Integer Sequences, 2003, http://www.research.att.com/˜njas/sequences.

[32] Takamizawa, K., Nishizeki, T., Saito, N.: Linear-time computability of combinatorial problems on series-parallel graphs, *J. ACM*, **29**(3), 1982, 623–641, ISSN 0004-5411.

[33] Thompson, K.: Regular expression search algorithm, *Communications of the ACM*, **11**(6), 1968, 410–422.

[34] Valdes, J., Tarjan, R. E., Lawler, E. L.: The recognition of Series Parallel digraphs, *STOC '79: Proceedings of the eleventh annual ACM symposium on Theory of computing*, ACM, New York, NY, USA, 1979.

[35] Yu, S.: Regular languages, in: *Handbook of Formal Languages* (G. Rozenberg, A. Salomaa, Eds.), vol. 1, Springer-Verlag, 1997.